

① Requirements

② Data Streaming


③ Intro. to IPC

Note Title

3/29/2007

Use Case Scenarios — play by play of what the user does & sees

Each line has 4 cols.

User Action	Immediate Feedback (transient)	Visible Result	Model State Change
click on line tool	line tool button is selected & cursor changes to 	—	—
press mouse button in canvas area	—	—	(remember the initial point of the line)
mouse drags to another point	see a line from original pt. to current position	—	(remember the 2nd point)

Line tool
Controller

more dragging

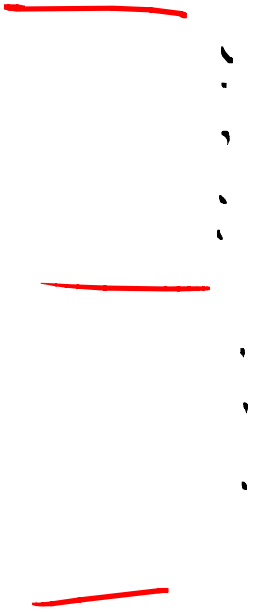
mouse release

line moves

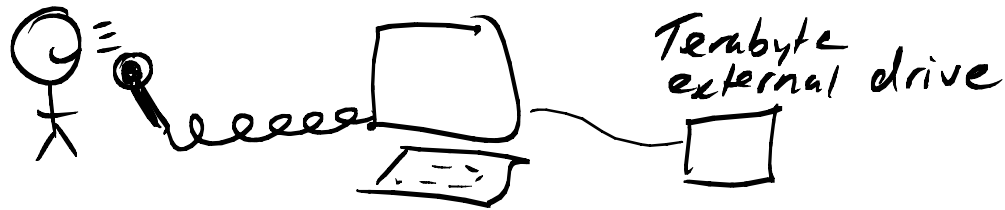
— (temporary line goes away)

—
final line is displayed

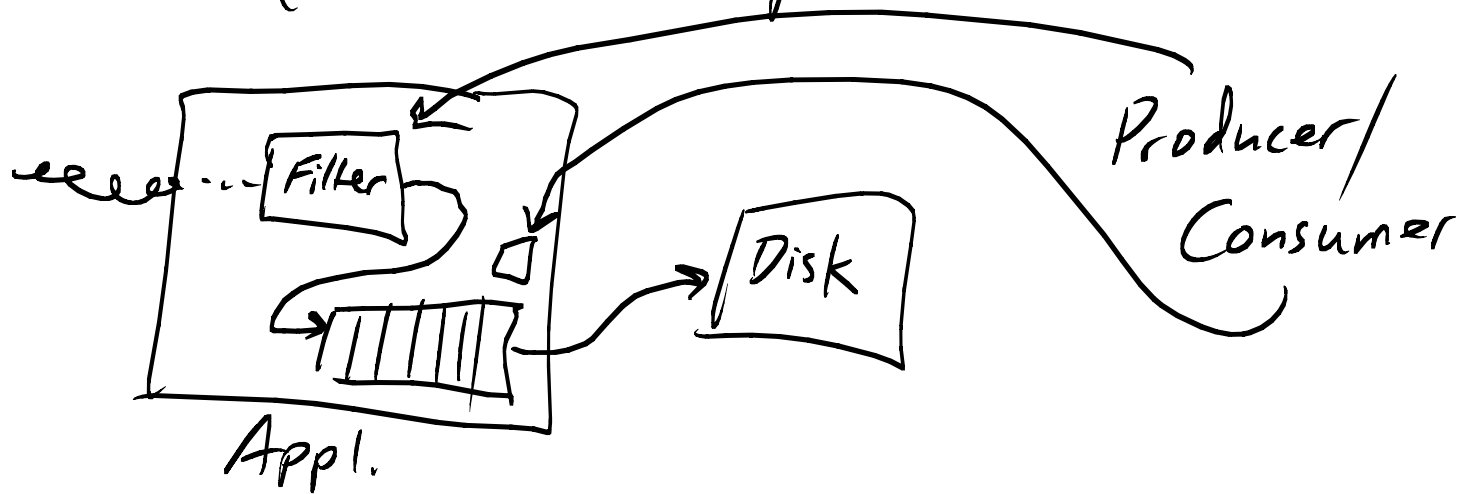
—
line is added to the shapes list



Data Streaming



Saving it all in memory & putting in a file at the end is not an option



throughput — $\frac{\text{how much data}}{\text{how much time}}$ $\frac{\text{bits}}{\text{sec}}$ $\frac{\text{songs}}{\text{hour}}$

latency — how long things take — wait time

disk latency — x ms

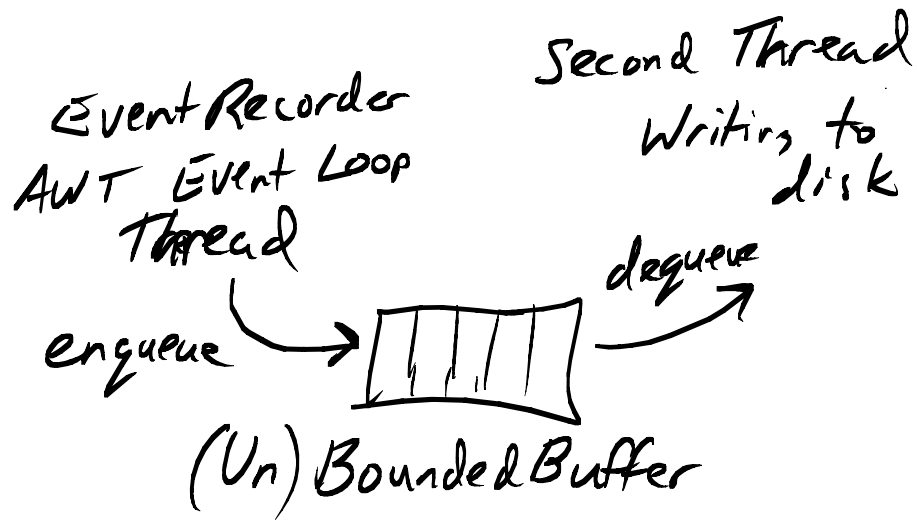
memory latency — y ns

mask latency by pipelining

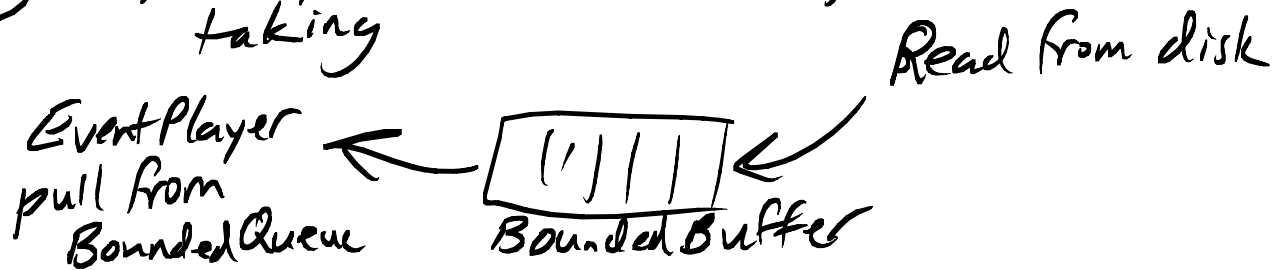
jitter — temporary delays due to emptying out the incoming buffer



① As user takes the survey,
stream user events to a file

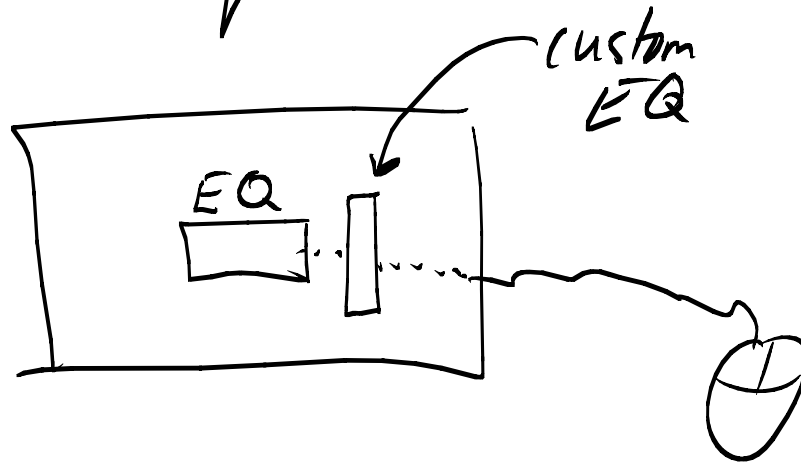


② Psychologist wants to analyze behavior during survey
taking



To avoid interference w/ mouse events during playback —

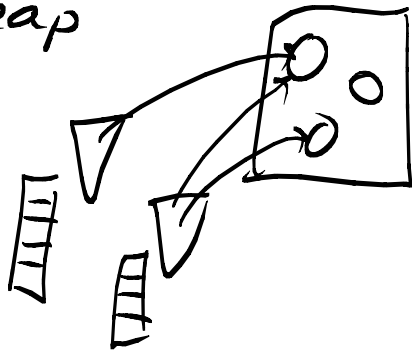
Let event player also put something on top of event queue



Interprocess Communication

Threads

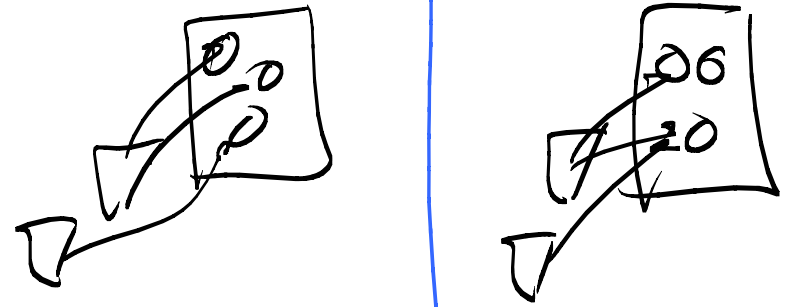
run within a single app.
with a single shared
heap



communication is through
shared state

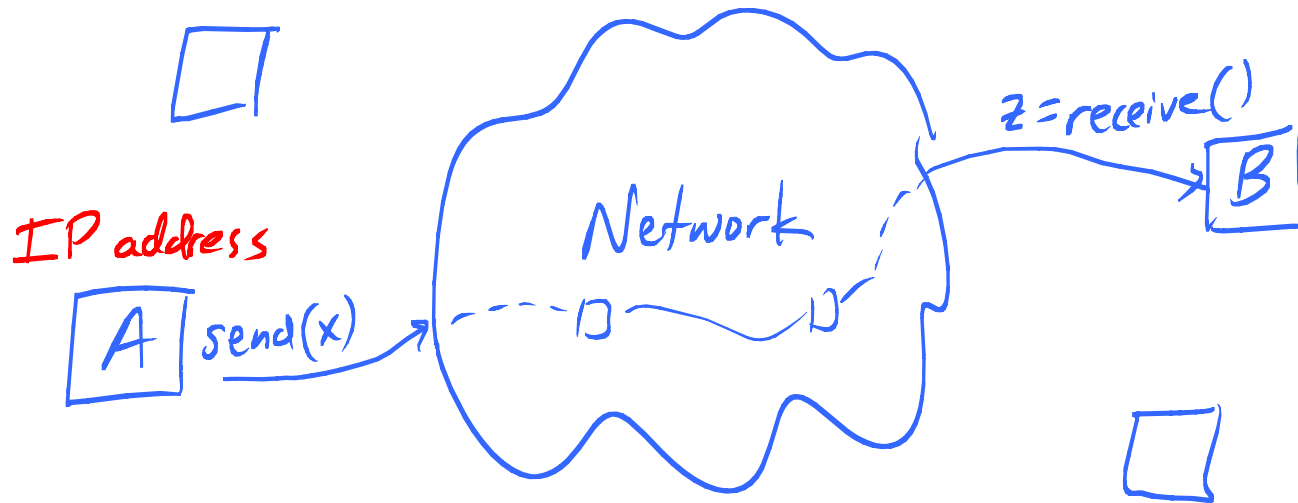
Processes

run in different address spaces
(protected by OS)



Communication

- ① O.S.-supported shared memory
- ② Message passing



Abstractions

- ① Streams
- ② Socket — network connection to another process

IP = Internet Protocol

agreed-upon way
of interacting

IP — datagrams (U.S. Postal Service on steroids)

"Best effort"

- message loss
 - garbled messages
- } possible

header payload check



TCP — Transmission Control Protocol TCP/IP

- ① handles message loss/duplication → reliable stream
- ② rate control (friendly to other network traffic) (FIFO)

TCP

message loss & reordering

Idea ① put a sequence # in header of each message

Receiver

1 3 4 2

reordering is a local operation


Sender

1 2 3 4

1 3 4 can't handle
missed messages locally

Idea ② retransmit missing packets

a. Negative Acknowledgment (NACK)

NACK 2 

b. Positive Acknowledgments (ACK)

 2

ACK 6 — I got 1, 2, 3, 4, 5, & 6

If sender sent 7, 8, 9, could retransmit

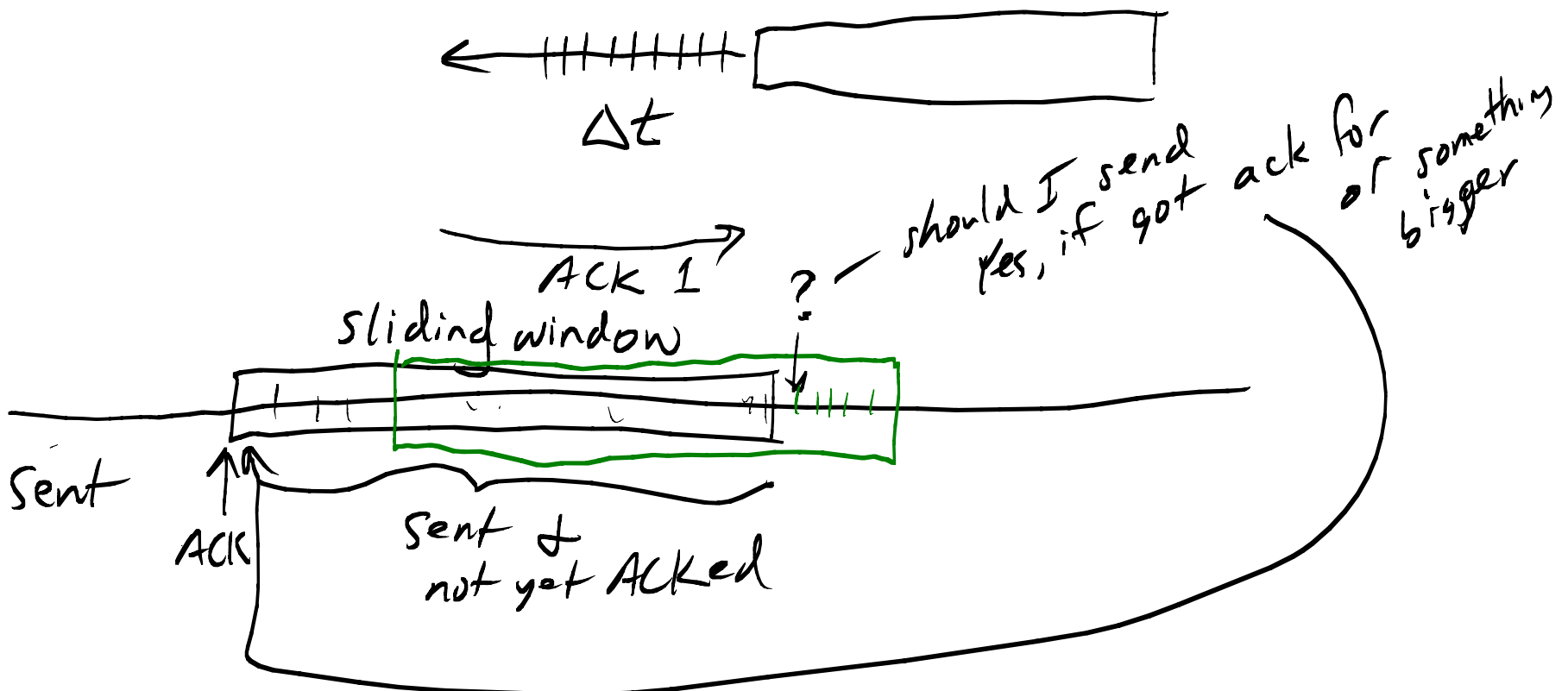
Protocol

Receiver

constantly ACK highest complete seq. #
reorder packets locally

Sender

send a fixed amount of stuff
& waits for ack before sending more



Window
Size



time

TCP slow start